# The influence of network topology on reverse-engineering of gene-regulatory networks

Procedia Computer Science

Volume 29, 2014, Pages 410–421

ICCS 2014. 14th International Conference on Computational Science

# The influence of network topology on reverse-engineering of gene-regulatory networks

Alexandru Mizeranschi, Noel Kennedy, Paul Thompson, Huiru Zheng, and Werner Dubitzky*

University of Ulster, Coleraine/Jordanstown, UK
w.dubitzky@ulster.ac.uk

**Abstract**

Modeling and simulation of gene-regulatory networks (GRNs) has become an important aspect of modern computational biology investigations into gene regulation. A key challenge in this area is the automated inference (reverse-engineering) of *dynamic, mechanistic GRN models* from time-course gene expression data. Common mathematical formalisms used to represent such models capture both the *relative weight* or strength of a regulator gene and the type of the regulator (activator, repressor) with a single model parameter. The goal of this study is to quantify the role this parameter plays in terms of the computational performance of the reverse-engineering process and the predictive power of the inferred GRN models. We carried out three sets of computational experiments on a GRN system consisting of 22 genes. While more comprehensive studies of this kind are ultimately required, this computational study demonstrates that models with similar training (reverse-engineering) error that have been inferred under varying degrees of a priori known topology information, exhibit considerably different predictive performance. This study was performed with a newly developed multiscale modeling and simulation tool called MultiGrain/MAPPER.

*Keywords:* Gene-regulation, automated model inference, rate law, structure parameters

## 1 Introduction

Systems biology refers to the quantitative analysis of the dynamic interactions among multiple components of a biological system and aims to understand the characteristics of a system as a whole [3]. It involves the development and application of system-theoretic concepts for the study of complex biological systems through iteration over mathematical modelling, computational simulation and biological experimentation. *Modeling and simulation* of gene-regulation networks is becoming an area of growing interest in systems biology research [13].

*Corresponding author

 Selection and peer-review under responsibility of the Scientific Programme Committee of ICCS 2014

doi:10.1016/j.procs.2014.05.037

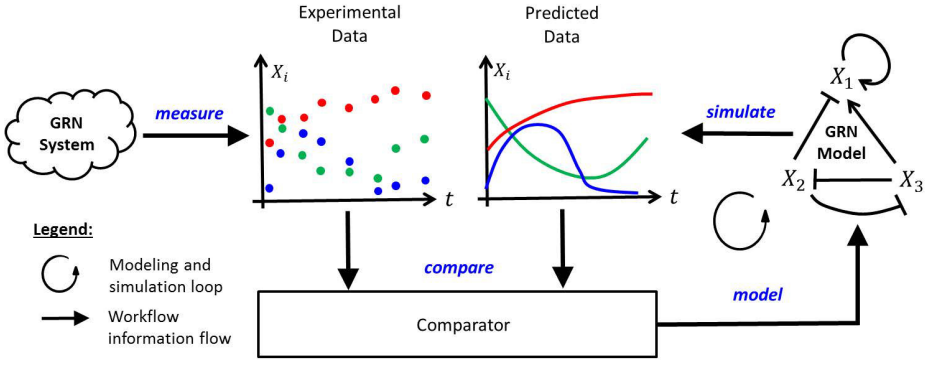


Figure 1: Automated inference of gene-regulatory networks from time-course gene expression data.

The regulation of genes and their products is at the heart of modern systems biology research. For instance, the understanding gene-regulatory processes in the context of diseases is increasingly important for therapeutic development. Cells regulate the expression of their genes to create functional gene products (RNA, proteins) from the information stored in genes (DNA) [2]. Gene regulation is a complex process involving the transcription of genetic information from DNA to RNA, the translation of RNA information to make protein, and the post-translational modification of proteins. Gene regulation is essential for life as it allows an organism to respond to changes in the environment by making the required amount of the right type of protein when needed. Developing *quantitative models of gene regulation* is essential to guide our understanding of gene regulation. The approaches considered in our study concentrate on an abstract conceptualization of gene-regulation networks that ignores intricate intermediate biological processes of cellular gene regulation, such as splicing, capping, translation, binding and unbinding [13].

As the amount of gene expression data is growing, researchers are becoming increasingly interested in the *automated inference* of quantitative dynamic, mechanistic gene-regulatory network *models* from time-course gene expression data [11, 13, 4, 17, 6, 7]. This inference process is depicted in Figure 1.

The modeling and simulation loop in the GRN model reverse-engineering workflow (Figure 1) keeps generating models which predict time-course gene expression data until the error between measured and predicted/simulated data falls below a pre-defined threshold. The quality of a reverse-engineered model is mainly determined by two factors:

1. **Predictive power**: The accuracy of the *predicted time-course response* for unseen stimulus/input data.

2. **Inferential power**: The accuracy of the reverse-engineered gene-regulatory *structure*.

Reverse-engineering high-fidelity GRN models is a long-standing problem [13]. Currently, some of the main challenges include a lack of sufficient amounts of time-course gene expression data and a lack of reverse-engineering algorithms and methods that are able to incorporate existing biological knowledge effectively. The data problem has two aspects:

1. Because of the high costs and other factors, the number of experiments performed on real GRN systems is often very small. In particular, experiments with different experimental

conditions applied to the same system are rare. As a consequence, there are normally only very few data sets available for modeling (model inference and model validation). Commonly there is only one data set for model inference, and none for model validation. Systematic studies on how many data sets are required are rare [11].

2. Many time-course gene expression experiments involve only a small number of time points; sometimes as low as two or three. Again, there is a lack of systematic studies that shed more light on this aspect.

More critical than the number of measured time points is to have stimulus-response data of more than one experimental condition [11], as a single-condition data set is unlikely to carry enough information to reveal the underlying network topology with sufficient accuracy.

A great deal of research on reverse-engineering dynamic, mechanistic GRN models involve only a small or moderate number (order of 5-10) of genes. Approaches with more than ten genes are still uncommon. One reason for this is the computational resources (processors) that are needed to reverse-engineer a GRN model – the more genes, the more computing power is needed. The objective of this study is to shed more light on the performance aspects of GRN model reverse-engineering. In particular, we focus on the issue on quantifying the computational complexity of the reverse-engineering process and the predictive power of the inferred models depending on the parameters that represent the network structure. This study was performed with a tool called MultiGrain/MAPPER (see Section 3), which we developed as part of the EU project MAPPER[1]. MultiGrain/MAPPER was designed for modelling and simulation of gene-regulation networks within a multiscale modelling framework.

The remainder of the paper is organized as follows: Section 2 describes the modeling and simulation of gene-regulatory networks. It discusses the important role of the network topology model parameter "omega" in the conceptualization (representation) of dynamic, mechanistic GRN models and in the model reverse-engineering process. Section 3 presents the computational experiments we have conducted to quantify the role of the "omega" parameter on the computational performance of the reverse-engineering process and the predictive performance of the resulting models. The results of these experiments are presented and discussed in Section 4. Section 5 concludes with some final remarks and an outlook on future work.

# 2 Modeling and simulation of gene-regulation networks

## 2.1 ODE-based GRN model representation

As time-course gene expression data are more readily available, mathematical modeling and computational simulation are becoming important tools for investigating the gene-regulatory structure and time-dependent changes of transcripts in GRNs [13, 7]. The most common strategy for modeling and simulating dynamic GRNs is based on nonlinear *ordinary differential equations* (*ODEs*) obtained from standard mass-balance[2] kinetic rate laws [3]. The ODEs in a GRN model relate changes in gene transcript concentrations to each other (and possibly to an external perturbations). Such GRN models consist of a differential equation for each of the genes in the network, describing the transcription rate of the gene as a function of the other genes (and of the external perturbations). The parameters of the equations have to be inferred from the time-course gene expression data set based on *a priori* knowledge. ODE GRN

[1] `http://www.mapper-project.eu`

[2] Mass-action kinetics assumes that the rate of an elementary reaction is proportional to the product of the concentrations of reacting species (reactants): $rate \propto [A][B]$ for $A + B \rightarrow C$.

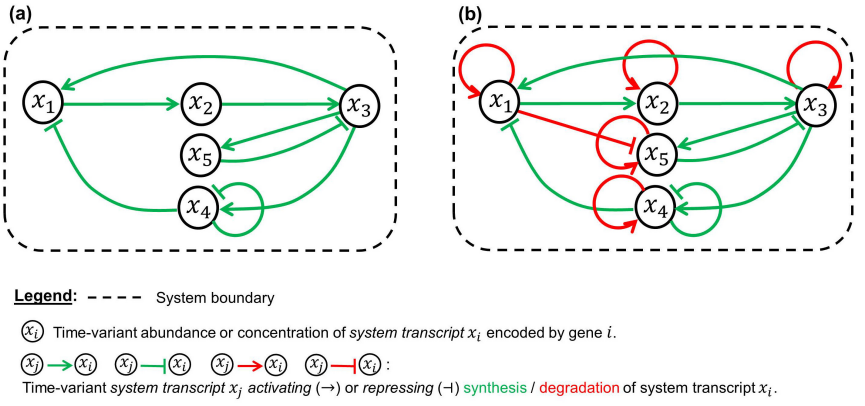


Figure 2: Illustration GRN network structure for 5-gene GRN system: (a) Synthesis-regulatory network structure. (b) Degradation-regulatory network structure.

models are similar to metabolic models that are formulated based on enzyme kinetics, where each rate law approximates a series of elementary chemical steps. Here, the rate laws are one level of complexity above that and represent a *series* of enzymatic steps. Because these rate laws combine mechanistic details into a small set of model parameters, they are sometimes referred to as "lumped" models. In a sense, these models are neither fully mechanistic nor purely phenomenological.

The equations in an ODE-based GRN model represent two gene-regulatory processes: transcript synthesis and transcript degradation. *Transcript synthesis* is a process that constructs (synthesizes) transcript molecules (messenger mRNA), and *transcript degradation* is a process that destroys (degrades) transcript molecules. The two processes are *regulated* based on the conditions within the cell and in its environment to ensure that the right amount of the right type of transcript is available when needed. Regulation means that a process is either *activated* or *repressed*. Synthesis activation increases the rate of transcript synthesis, and synthesis repression decreases it. Degradation activation increases the rate of transcript degradation, and degradation repression decreases it. Both processes combine to regulate the *total amount of transcript* in a cell. Mathematically, a GRN model is defined by a set of coupled ODEs. There is one ODE for each gene in the GRN.

Eq. (1a) illustrates a generic ODE-based GRN model defining the total rate of transcript change of a GRN system consisting of $i = 1, 2, ..., n$ genes. Eq. (1b) provides a simplified notation by making the time variable $t$ "implicit". We will use this simplified form in the remainder of this text.

$$\frac{dx_i}{dt}(t) = \Omega_i(S_i, x_1(t), x_2(t), ..., x_n(t), t) - \Delta_i(D_i, x_1(t), x_2(t), ..., x_n(t), t) \tag{1a}$$

$$\frac{dx_i}{dt} = \Omega_i(S_i, x_1, x_2, ..., x_n) - \Delta_i(D_i, x_1, x_2, ..., x_n) \tag{1b}$$

where

- $x_i, x_j \in \{x_1, x_2, ..., x_n\}$: *Transcript concentration* of genes $i$ and $j$, respectively, at time $t$;
- $dx_i/dt$: *Total rate of* $x_i$ *change* at time $t$;

- $\Omega_i(\cdot)$: The *rate of synthesis* of transcript $x_i$ at time $t$;
- $\Delta_i(\cdot)$: The *rate of degradation* of transcript $x_i$ at time $t$;
- $S_i, D_i$: Constant *parameters* governing $\Omega_i(\cdot)$ and $\Delta_i(\cdot)$, respectively;
- $t$: Time.

Notice, while the total rate of transcript change, $dx_i/dt$, may be negative at any given time, neither the rate of transcript synthesis nor the rate of transcript degradation can be negative!

An important aspect in ODE-based GRN models is the representation of the regulatory network *structure*. The structure is determined by the *nature* (activator, repressor, no regulation) and *relative strength* or weight of influence of transcript $x_j$ on transcript $x_i$. The gene-regulatory structure of a GRN is usually conceptualized as a *network* (or graph) consisting of *nodes* (or vertices) and *links* (or edges) between nodes [1]. The nodes represent *genes* or *transcripts*, and the links represent gene-regulatory or transcript-regulatory influences. Mathematically, such networks are described as *edge-labeled directed graphs*.

Usually, visualizations of GRN graphs depict only the influences (edges) regulating transcript *synthesis*. Figure 2a illustrates such a synthesis-regulatory graph structure based on a 5-gene GRN. Since there are only two types of synthesis regulation (synthesis activation and repression), the edge "labels" of a synthesis-regulatory graph are normally visualized by drawing arrows with two different heads, "→" and " ⊣ " as illustrated in Figure 2a. A directed graph (edge-labeled or not) allows more than one edge joining the same vertex, but more than one edge going in the same direction between two vertices is not allowed. This is consistent with our conceptualization of GRNs which requires that a regulator gene cannot simultaneously activate and repress transcript synthesis of the same target gene. Figure 2b shows the same GRN system depicted in Figure 2a, but with the addition of the regulatory structure governing the process of transcript *degradation*. In general, the degradation-regulatory structure of GRNs is also represented as an edge-labeled directed graph. Even though most, if not all, GRN models represent transcript degradation in some form or another, the visualizations of the regulatory network structure do not usually show the degradation-regulatory influences, because degradation is often modeled as a uniform process that does not involve complex regulatory influences between distinct genes.

We can appreciate how ODE-based GRN models represent the network structure of a GRN system when we replace the generic ODE GRN model shown in Eq. (1) by a concrete ODE rate law. In this study we chose a widely used rate model called *artificial neural network* (*ANN*) [19], which is defined by Eq. (2).

$$\frac{dx_i}{dt} = \hat{\alpha}_i \frac{1}{1 + exp(\gamma_i - \sum_{j}^{n} \omega_{ij}\ x_j)} - \beta_i x_i\ , \tag{2}$$

where

- $x_i, x_j \in \{x_1, x_2, ..., x_n\}$: *Transcript concentration* of genes $i$ and $j$, respectively, at time $t$, where $n$ is the total number of genes in the GRN system;
- $dx_i/dt$: *Total rate of* $x_i$ *change* at time $t$;
- $\hat{\alpha}_i \in \mathbb{R}^+$: *Maximal synthesis rate* of transcript $x_i$;
- $\omega_{ij} \in \mathbb{R}$: *Type* of synthesis regulation of transcript $x_i$ by $x_j$, such that
  $\omega_{ij} > 0$: *synthesis activation* of $x_i$ by $x_j$;
  $\omega_{ij} < 0$: *synthesis repression* of $x_i$ by $x_j$;

$\omega_{ij} = 0$: *no synthesis regulation* of $x_i$ by $x_j$.

- $|\omega_{ij}| \in \mathbb{R}_0^+$: *Relative weight* of synthesis-regulatory influence of $x_j$ on $x_i$;
- $\gamma_i \in \mathbb{R}$: *Sensitivity* of $x_i$ synthesis to the combined influence $\sum_j \omega_{ij}\ x_j$ of the regulators; and
- $\beta_i \in \mathbb{R}^+$: *Degradation rate constant* modulating degradation rate of $x_i$.

The ANN rate law is inspired by research on artificial neural neurons [14]. In this formulation, the transcript synthesis rate $\Omega_i(\cdot)$ (first term in Eq. (2)) is determined by summing up the weighted influences of all transcripts in the system with an exponential function that produces sigmoidal kinetics. Because $1/(1 + exp(\cdot)) \in [0, 1]$, the constant $\hat{\alpha}_i$ represents the *maximal* rate at which $x_i$ can be synthesized. And the transcript degradation rate $\Delta_i(\cdot)$ (second term in Eq. (2)) is modeled as being directly proportional to the concentration of the transcript $x_i$ itself: $\Delta_i(\cdot) = \beta_i x_i$.

Two other well-known ODE-based formalisms that have been used to model GRN systems are the *Hill* [9] and the *synergistic-system* [16] rate laws.

## 2.2 Reverse-engineering of GRN models from time-course data

Once one has chosen a formalism to represent a GRN, one needs to determine *concrete values* of the model parameters – the parameters that describe the network structure, and the parameters that represent other system aspects. These values could be determined manually based on available knowledge about the underlying GRN system, or automatically based on available experimental data from the GRN system.

***ReverseEngineerModel***
**Input**: $M \leftarrow$ Model equations; $L \leftarrow$ Parameter limits; $G \leftarrow$ Network topology
**Input**: $D \leftarrow$ Training data; $\varepsilon \leftarrow$ Error threshold
**Output**: $P \leftarrow$ Parameter values; $E \leftarrow$ Training error;
$S \leftarrow$ Simulation data `(* Initialize simulation data *)`
$E \leftarrow \infty$ `(* Initialize training error *)`
**repeat**
  $P \leftarrow Optimize(L, E)$ `(* Generate parameter values with optimizer *)`
  $S \leftarrow SolveODE(M, P, D)$ `(* Solve model equations with ODE solver *)`
  $E \leftarrow Error(S, D)$ `(* Determine training error *)`
**until** $E < \varepsilon$ ;

**Algorithm 1:** Basic reverse-engineering algorithm. The network topology, $G$, is an optional input. In this study, we experiment with various degrees of known network topology.

The automated approach whose basic workflow is depicted in Figure 1 requires a suitable algorithm to estimate the model parameters. A pseudo code of the a general reverse-engineering algorithm is shown in Algorithm 1. The algorithm is called "general" because it shows only key steps and does not make explicit certain details, e.g. keeping track of and memorizing the current best solutions.

Algorithm 1 illustrates the basic steps in reverse-engineering an ODE GRN model from time-course gene expression data. The main loop contains the critical steps of the algorithm: (1) An *optimizer* algorithm that generates candidate model parameter values by attempting to minimize the training error, $E$. (2) An *ODE solver* component that numerically integrates the model equations using the initial values of the time series in the training data set, $D$. (3) A

component that computes the *simulation error*, $E$, based on the time-course gene expression data in the training data set, $D$, and the predicted or simulated data, $S$, determined by the ODE solver. In terms of computational effort, the ODE solver step in the algorithm accounts for approximately for 80% of the total computing time of Algorithm 1. The reverse-engineering process terminates, when the training error drops below the pre-defined error threshold $\varepsilon$. Usually, a maximal number of iterations is defined to make sure that the algorithm terminates in reasonable time (this is not shown in Algorithm 1).

Once a "final" model has been determined with Algorithm 1, this model still needs to be validated before it is accepted. There are a variety of ways to validate a system dynamics model [5]. A common approach, adopted in this study, is to simulate the response data, $S(V)$, to an initial condition defined in a separate, independent validation data set, $V$, and then calculate the *validation error* between $S(V)$ and $V$. If this validation error is within acceptable boundaries, the model is said to be validated.

### 2.3 The trouble with "omega"

The omega parameter in the ANN rate law[3] plays a crucial role, because it represents two distinct biological concepts simultaneously. On one hand, it defines the nature of synthesis regulation between two genes $i$ and $j$ – if it is positive, then $j$ activates $i$, if it is negative, $j$ represses $i$, and if it is zero, $j$ does not regulate $i$ . This could be viewed as a logical or qualitative role of $\omega_{ij}$. On the other hand, its absolute value $|\omega_{ij}|$ defines the relative strength or weight of a regulator gene $j$ on the target gene $i$. An important consequence of such a continuous representation of a discrete concept (network topology) is that a reverse-engineering algorithm like the one discussed above has a tendency to infer fully connected network structures. Fully connected structures are difficult to interpret at best, or totally meaningless at worst. Future reverse-engineering algorithms need to employ heuristics and/or background knowledge to limit the number of inferred regulatory influences.

The number of parameters that need to be determined by the reverse-engineering procedure depends on the rate law. Critically, the number of parameters grows with the number of genes in the underlying GRN system. Since the $\omega_{ij}$ parameter is linked to synthesis-regulatory interactions, the number of omega parameters grows with the number, $n$, of genes in the GRN system. For a GRN system consisting of $n$ genes, there is a maximum of $n^2$ synthesis-regulatory interactions (and $n^2$ degradation-regulatory interactions). For the ANN rate law defined by Eq. (2), the total number of parameter values to be estimated is $(n+3)n$, since we have $\hat{\alpha}_i, \gamma_i, \beta_i, \omega_{i1}, \omega_{i2}, ..., \omega_{in}$ for each gene $i$.

The goal of this study is to quantify the role the omega parameters play in terms of the computational performance of the reverse-engineering algorithm and the predictive performance of the reverse-engineered GRN models.

## 3 Experiments and implementation

### 3.1 Experiments

To quantify the influence of the omega parameter on the complexity of the reverse-engineering computations and the fidelity of the resulting models, we have designed three sets of experiments based on the yeast cyclin GRN system that consists of 22 genes [18]. In yeast and other

[3]The Hill [9] and synergistic-system [16] rate laws have an equivalent parameter.

organisms, the cell cycle is controlled by the activity of the protein family of cyclins and cyclin-dependent kinases that are periodically expressed during cell cycle. Because these proteins are conserved among different species, insights into how this network regulates the abundance of its proteins is of importance well beyond the yeast species. However, in this study we did not use the exact yeast GRN as reported by To and Vohradský [18], instead, we created artificial variants consistent with the basic characteristics (number and type of synthesis regulators) of the yeast GRN system.

In order to investigate the influence of topology information encoded by the omega parameters, we conducted three sets of reverse-engineering experiments based on the ANN rate law defined by Eq. (2):

1. **Unknown network topology**: The topology of the GRN network is not known, hence all $n^2 = 22^2 = 484$ omega parameters plus $3 * n = 3 * 22 = 66$ non-omega parameters need to be estimated. In total, 550 parameters.

2. **Fully known network topology**: The nature of all 64 synthesis regulators (34 activators and 30 repressors) in the GRN is known. This means that 420 omega parameters are fixed at a value of zero, and the values of the remaining 64 omega-parameters need to be estimated within a signed range: $(0, +\omega_{max}]$ for activators and $[-\omega_{min}, 0)$ for repressors. In addition, all 66 non-omega parameters need to be estimated.

3. **Partially known network topology**: 32 of the 64 synthesis regulators are randomly identified as known (using the corresponding limits as in case (2) above). This means that 66 non-omega and 452 omega parameters need to be estimated.

We created a reference GRN systems with the JAGN [12] artificial gene network generation tool. From this, we generated two independent data sets (each with a different initial condition). One data set, the *training data set*, was used for model inference (reverse-engineering), and the other, the *validation data set*, for model validation. For each training data set, we reverse-engineered 20 models[4] and validated each against the corresponding validation data set. The reverse-engineering simulation error threshold was set to $\varepsilon = 0.01$, and each of the 20 models was guaranteed to have a training error, $E$, below this threshold: $E < \varepsilon$. We employed the *root mean squared error* (*RMSE*) to compute $E$.

The optimization step of the reverse-engineering algorithm was realized with *particle swarm optimization* (*PSO*). PSO is a population-based stochastic optimization technique that is particularly useful for estimating the values of continuous parameters [10]. It is inspired by the flocking or schooling behavior of animals and shares many similarities with evolutionary computation techniques, such as genetic algorithms. Similar to genetic algorithms, the PSO is initialized with a population of random solutions, called *particles*. The particles "move" through the search space in a goal-directed fashion dependent on their "fitness". In each iteration, the position and velocity of a particle is updated based on its fitness and its relative position to neighboring particles. Our experiments adopted a multi-island or *multi-swarm PSO* approach. Similar to genetic algorithms, a multi-swarm PSO spreads its entire particle population over multiple islands or swarms. During optimization, a few particles are allowed to cross between swarms to facilitate "genetic" diversity. The multi-swarm PSO is assumed to have better properties (than the single-swarm variant) in terms of avoiding convergence to local optima. Our reverse-engineering experiments were performed with a multi-swarm set-up consisting of 10

[4]Because of the stochastic nature of the optimizing step in Algorithm 1, each reverse-engineered model is different. Hence, we created 20 replicas to obtain robust statistics.

swarms, each swarm consisting of 100 particles. The fitness in our experiments was determined by the training error, $E$, with a threshold of $\varepsilon = 0.01$.

All experiments were performed with our own GRN modeling and simulation tool MultiGrain/MAPPER (see next section). The execution environment was preserved for all scenarios. It consisted of a high-performance computing cluster consisting of 1088 nodes (12 CPU cores per node) based on an Intel Xeon architecture with either 16 and 24 GB of memory per node, respectively. The cluster was made available by the Cyfronet Academic Computer Centre in Krakow, Poland.

### 3.2 MultiGrain/MAPPER: a GRN modeling and simulation tool

As part of the European MAPPER project (see Footnote 1), we developed a Java-based software called *Multiscale Gene Regulation Modeling Tool* (*MultiGrain/MAPPER*). MultiGrain/MAPPER is still under active development. One of the main design goals of MultiGrain/MAPPER is to facilitate distributed multiscale modeling and simulation of gene regulation systems and processes. MultiGrain/MAPPER supports the Systems Biology Mark-up Language (SBML) and the Systems Biology Results Mark-up Language (SBRML). The ODE solver in MultiGrain/MAPPER uses Michael Thomas Flanagan's Java Library[5]. A core feature of MultiGrain/MAPPER is a multi-swarm implementation of particle swarm optimization [10]. The multi-swarm PSO feature allows us to map the optimization process to multiple and possibly distributed processors and machines. We use the MAPPER *MUltiScale Coupling Library and Environment* (*MUSCLE*) library [8] to implement the communication channels between MultiGrain/MAPPER swarms and the GRN particles. MUSCLE was originally developed around a complex automata model of multiscale systems, which provides a framework for linking (coupling) single-scale models to create complex multiscale models.

## 4 Results and discussion

The results of our experiments are summarized in Table 1. The table shows the average number of *PSO iterations* performed, the average *wall time* taken, and the average *validation error* obtained over 20 reverse-engineering experiments per condition (unknown, partially known, and fully known network topology). Recall, the *training error* of all 20 models for each topology conditions was less than $\varepsilon = 0.01$.

| Topology | ValError | TotalTime [sec] | Iterations |
|---|---|---|---|
| **notop** | 0.063 ± 0.083 | 174.80 ± 196.09 | 16.70 ± 21.67 |
| **part** | 0.042 ± 0.063 | 128.55 ± 126.63 | 12.25 ± 14.53 |
| **top** | 0.014 ± 0.003 | 54.25 ± 66.35 | 5.95 ± 5.31 |

Table 1: Results from three sets of reverse-engineering experiments. Each cell shows the mean value and associated standard deviation for $n = 20$ repeats. (*notop*=unknown network topology; *part*=partially known network topology; *top*=fully known network topology)

The boxplots in Figures 3 and 4 visualize all results for the validation error of the reverse-engineered GRN models and computation time (wall time) of the reverse-engineering processes.

Consider the validation error results in Figure 3 and Table 1. First, the boxplots clearly show a decrease in variability of the validation error as we go from unknown topology (*notop*),

[5]Michael Thomas Flanagan's Java Scientific Library: `http://www.ee.ucl.ac.uk/~mflanaga/java/`

to partially known topology (*part*), and through to fully known topology (*top*). The variation of the *mean* of the validation errors is better conveyed by the last column of Table 1. Indeed, the mean validation error of the 20 models that have been reverse-engineered with fully known topology is about 4.5 times lower than that of the models obtained with unknown topology. This is remarkable because *all* 60 reverse-engineered models have a training error of less than the threshold $\varepsilon = 0.01$.

Consider the total time results in Figure 4 and Table 1. Both the boxplot and the data in the TotalTime column of the table suggest that the variation of the total reverse-engineering wall times is in the order of the average total time of the corresponding topology condition. Somewhat surprisingly, the mean total wall time for the unknown topology experiments is only 3.22 seconds longer than that for the experiments with the fully known topology. This is surprising, because for the experiments with unknown topology there are approximately 3.75 times more parameters to be estimated than in the experiments with fully known topology. This suggests a good scalability of the implementation of the reverse-engineering algorithm.

The total wall time data correlates well with the number of iterations (column with header Iterations in Table 1) performed by the reverse-engineering process. Therefore, we do not further discuss these data.

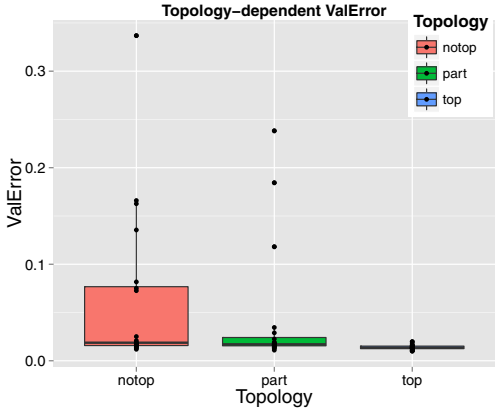


Figure 3: Validation error.

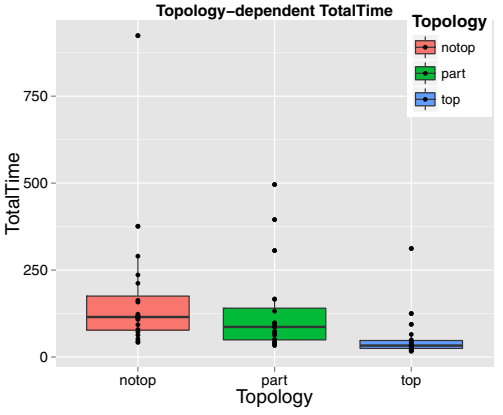


Figure 4: Reverse-engineering total time.

# 5 Conclusions and future work

Automated inference of gene-regulation models from time-course gene expression data is a long-standing problem [13]. Such models are useful, provided they perform well on unseen data (predictive power) and that their regulatory structure is close to the real structure of the underlying GRN *system* (inferential power). The inference process is a challenging task because of the lack of sufficient data. In particular, data obtained from multiple experiments under different experimental conditions is lacking. Such data is likely to reveal more details of the underlying system [11].

Common rate law model formalisms employ a single parameter per (potential) synthesis regulator to express the regulator's *nature* (activator, repressor, no regulator) as well as the relative influence weight or *strength* of the regulator [3, 15, 19, 16, 9]. In trying to cover

both concepts, these rate laws are prone to infer *fully connected* network structures. This is a big problem, as such structures cannot be interpreted in a meaningful way. When we separate structure from relative strength in the inference process, we would like to have a better idea how big an influence the structure has on the inference process, both in terms of computational complexity and model fidelity. In this study, we explored this question based on a GRN system consisting of 22 genes. We found that for all models that fitted the training data well (training error: $E < 0.01$), only those whose structure is close to the structure of the system that generated the data perform well on the independent validation data set (validation error). We were able to quantify the difference based on the three experiments (with different topology assumptions) we performed. We were also able to quantify the computing time for the different topology configurations. One of the contributions this study makes is to generate quantities for these aspects. We note that our implementation of the reverse-engineering Algorithm 1 scales roughly linearly with number of structure parameters (this is not discussed in detail here for space limitation reasons). Our results underline the importance of having independent validation data sets generated from the same system from which the models are constructed.

While we believe that the result of our experiments are important, clearly more systematic computational studies are needed to understand the type of GRN model inference investigated in this study. We will continue to work in this area in the future.

## Acknowledgments

This research received funding from the MAPPER EU-FP7 project (grant no. RI-261507) and was supported in part by PL-Grid infrastructure.

## References

[1] T. Aittokallio and B. Schwikowski. Graph-based methods for analysing networks in cell biology. *Briefings in Bioinformatics*, 7(3):243–255, 2006.

[2] B. Alberts, A. Johnson, J. Lewis, M. Raff, K. Roberts, and P. Walter. *Molecular biology of the cell*. Garland Science, New York, 4 edition, 2002.

[3] U. Alon. *An Introduction to systems biology: Design principles of biological circuits*. CRC Press, Taylor & Francis Group, London, 2006.

[4] S.G. Baker and B.S. Kramer. Systems biology and cancer: Promises and perils. *Progress in Biophysics and Molecular Biology*, 106(2011):410–413, 2011.

[5] Y. Barlas. Model validation in systems dynamics. In *International Systems Dynamics Conference*, pages 1–10, 1994.

[6] I. Cantone, L. Marucci, F. Iorio, M.A. Ricci, V. Belcastro, M. Bansal, S. Santini, di Bernardo M., di Bernardo D., and Cosma M.P. A yeast synthetic network for in vivo assessment of reverse-engineering and modeling approaches. *Cell*, 137:172–181, 2009.

[7] K.-H. Cho, S.-M. Choo, S.H. Jung, J.-R. Kim, H.-S. Choi, and J. Kim. Reverse engineering of gene regulatory networks. *IET Systems Biology*, 1(3):149–163, 2007.

[8] J. Hegewald, M. Krafczyk, J. Tölke, A. Hoekstra, and B. Chopard. An agent-based coupling platform for complex automata. In M. Bubak, GD. Albada, J. Dongarra, and P.M.A. Sloot, editors, *Computational Science – ICCS 2008*, volume 5102 of *Lecture Notes in Computer Science*, pages 227–233. Springer Berlin Heidelberg, 2008.

[9] A.V. Hill. The possible effect of the aggregation of the molecules of hæmoglobin. *Journal of Physiology*, 40:iv–vii, 1910.

[10] J. Kennedy and R Eberhart. Particle swarm optimization. In *Proceedings of IEEE International Conference on Neural Networks*, volume IV, pages 1942–1948, 1995.

[11] N. Kennedy, A. Mizeranschi, P. Thompson, H. Zheng, and W. Dubitzky. Reverse-engineering of gene regulation models from multi-condition experiments. In *IEEE Symposium Series on Computational Intelligence 2013 (SSCI 2013)*, pages 112–119, Singapore, 2013.

[12] F.M. Lopes, R.M. Cesar, and L.da.F. Costa. Gene expression complex networks: Synthesis, identification, and analysis. *Journal of Computational Biology*, 18(10):1353–1367, 2011.

[13] D. Marbach, J.C. Costello, R. Küffner, N.M. Vega, R.J. Prill, D.M. Camacho, K.R. Allison, The DREAM5 Consortium, M. Kellis, J.J. Collins, and G. Stolovitzky. Wisdom of crowds for robust gene network inference. *Nature Methods*, 9:796–804, 2012.

[14] W. McCulloch and W. Pitts. A logical calculus of ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 5(4):115–133, 1943.

[15] P. Mendes, W. Sha, and K. Ye. Artificial gene networks for objective comparison of analysis algorithms. *Bioinformatics*, 19(suppl 2):ii122–ii129, 2003.

[16] M.A. Savageau. Introduction to s-systems and the underlying power-law formalism. *Mathematcial and Computer Modelling*, 11:546–551, 1988.

[17] M.T. Swain, J.J. Mandel, and W. Dubitzky. Comparative study of three commonly used continuous deterministic methods for modeling gene regulation networks. *BMC Bioinformatics*, 11(1):459, 2010.

[18] C.C. To and J. Vohradský. Measurement variation determines the gene network topology reconstructed from experimental data: A case study of the yeast cyclin network. *The FASEB Journal*, 24(9):3468–3478, May 2010.

[19] J. Vohradský. Neural network model of gene expression. *The FASEB Journal*, 15(3):846–854, 2001.